@BeanPostConstruct

Spring static factory-method example

------------------------------------

In Spring framework, if you want to create a bean by invoking a static factory-method, whose purpose is to encapsulate the object-creation process in a static method then you could use factory-method attribute.

Static factory method example

If you want to create different EmployeeDTO objects based on it’s designation using a static factory method, you can do it like below example.

EmployeeDTO.java

public class EmployeeDTO {

private Integer id;

private String firstName;

private String lastName;

private String designation;

//Setters and Getters

@Override

public String toString() {

return "Employee [id=" + id + ", firstName=" + firstName

+ ", lastName=" + lastName + ", type=" + designation + "]";

}

}

Create static factory method.

EmployeeFactory.java

public class EmployeeFactory {

public static EmployeeDTO createEmployeeOfType(String type)

{

if ("manager".equals(type) || "director".equals(type))

{

EmployeeDTO employee = new EmployeeDTO();

employee.setId(-1);

employee.setFirstName("dummy");

employee.setLastName("dummy");

//Set designation here

employee.setDesignation(type);

return employee;

}

else

{

throw new IllegalArgumentException("Unknown product");

}

}

}

Use factory-method attribute for creating the beans.

applicationContext.xml

<bean id="manager" class="com.howtodoinjava.demo.factory.EmployeeFactory"

factory-method="createEmployeeOfType">

<constructor-arg value="manager" />

</bean>

<bean id="director" class="com.howtodoinjava.demo.factory.EmployeeFactory"

factory-method="createEmployeeOfType">

<constructor-arg value="director" />

</bean>

Demo

Let’s test above static factory-method configuration.

TestSpringStaticFactoryMethod.java

public class TestSpringStaticFactoryMethod

{

@SuppressWarnings("resource")

public static void main(String[] args) throws Exception

{

ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

EmployeeDTO manager = (EmployeeDTO) context.getBean("manager");

System.out.println(manager);

EmployeeDTO director = (EmployeeDTO) context.getBean("director");

System.out.println(director);

}

}

Watch the output in console.

Console

Employee [id=-1, firstName=dummy, lastName=dummy, type=manager]

Employee [id=-1, firstName=dummy, lastName=dummy, type=director]

------------------------------------------------------------------------

Spring Bean Creation – Static Factory Method & Instance Factory Method

In Spring we can create bean using Spring FactoryBean, FactoryBean is an interface and we need to give implementations for the methods in it.

If you don’t want to go by that methodology but still want Java Factory Pattern to be implemented then we can go for Static Factory Method and

Instance Factory Method.

The client who requests for an object can simply make a call to the factory method which we have defined without knowing about the creation detail.

We will be using factory-method and factory-bean attribute in our configuration for the Injection of Bean, through the below spring factory pattern

example lets learn more about it.

1.factory-method: factory-method is the method that will be invoked while injecting the bean. It is used when the factory method is static

2.factory-bean: factory-bean represents the reference of the bean by which factory method will be invoked. It is used if factory method is non-static.

Spring Bean Creation – Static Factory Method

Folder Structure:

1.Create a new Java Project “SpringCoreTutorial” and create a package for our src files “com.javainterviewpoint“

2.Add the required libraries to the build path. Java Build Path ->Libraries ->Add External JARs and add the below jars.

“

commons-logging-1.2.jar

spring-beans-4.2.4.RELEASE.jar

spring-core-4.2.4.RELEASE.jar

spring-context-4.2.4.RELEASE.jar

spring-expression-4.2.4.RELEASE.jar

3.Create the Java classes Employee.java,EmployeeFactory.java and EmployeeLogic.java under com.javainterviewpoint.springfactory folder.

4.Place our configuration file SpringConfig.xml in the src directory

Employee.java

package com.javainterviewpoint.springfactory;

public class Employee

{

private String name;

private String age;

private String designation;

public Employee()

{

super();

}

public Employee(String name, String age, String designation)

{

super();

this.name = name;

this.age = age;

this.designation = designation;

}

public String getName()

{

return name;

}

public void setName(String name)

{

this.name = name;

}

public String getAge()

{

return age;

}

public void setAge(String age)

{

this.age = age;

}

public String getDesignation()

{

return designation;

}

public void setDesignation(String designation)

{

this.designation = designation;

}

@Override

public String toString()

{

return "\*\*\*Employee Details\*\*\*\n Name :" + name +"\n "

+ "Age : " + age + "\n Designation : " + designation;

}

}

Employee class is a simple POJO consisting of the getters and setters of the properties name, age and designation

Spring Factory Pattern Example – EmployeeFactory.java

package com.javainterviewpoint.springfactory;

public class EmployeeFactory

{

private EmployeeFactory()

{

}

public static Employee createEmployee(String designation)

{

Employee emp = new Employee();

if ("manager".equals(designation))

{

emp.setName("Manager JavaInterviewPoint");

emp.setAge("111");

emp.setDesignation(designation);

}

else if("seniormanager".equals(designation))

{

emp.setName("SeniorManager JavaInterviewPoint");

emp.setAge("222");

emp.setDesignation(designation);

}

else

{

throw new RuntimeException();

}

return emp;

}

}

EmployeeFactory is the factory class, which has a private constructor and the only way we can create object for the “EmployeeFactory” class is through the static method createEmployee(). We will be passing the value to our designation property from the spring bean property file.

SpringConfig.xml

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:context="http://www.springframework.org/schema/context"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-3.0.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context-3.0.xsd">

<context:annotation-config></context:annotation-config>

<bean id="employee" class="com.javainterviewpoint.springfactory.EmployeeFactory"

factory-method="createEmployee">

<constructor-arg value="seniormanager"></constructor-arg>

</bean>

</beans>

•In our spring bean property file we have created a bean for our EmployeeFactory class and have mentioned the factory-method as “createEmployee”.

•We have used Spring’s constructor injection to inject value to the argument “designation” of our createEmployee() method. You may wonder why ? As per the Official Spring documentation section 5.4.1 Arguments to the static factory method can be supplied through <constructor-arg>, exactly the same as if a constructor had actually been used.

EmployeeLogic.java

package com.javainterviewpoint.springfactory;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class EmployeeLogic

{

public static void main(String args[])

{

//Read the Configuration file using ApplicationContext

ApplicationContext applicationContext =

new ClassPathXmlApplicationContext("SpringConfig.xml");

//Get the Employee class instance

Employee employee = (Employee)applicationContext.getBean("employee");

System.out.println(employee);

}

}

•In our EmployeeLogic class we have read the Configuration file(SpringConfig.xml) and get all the bean definition through ApplicationContext

•Get the Employee Class instance by calling the getBean() method over the context created.

•Since we have passed the value to the designation argument as “seniormanager” through <constructor-arg> it will be printing the details of the SeniorManager

Output:

Once we run the EmployeeLogic class we will be getting the below output

Spring Bean Creation - Static Factory Method